UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/781,307	02/18/2004	John G. Beltran	G&C 30566.296-US-U1	4847

55895 7590 07/19/2011

GATES & COOPER LLP
HOWARD HUGHES CENTER
6701 CENTER DRIVE WEST, SUITE 1050
LOS ANGELES, CA 90045

EXAMINER	
ABDUL-ALI, OMAR R	
ART UNIT	**PAPER NUMBER**
2172	
MAIL DATE	**DELIVERY MODE**
07/19/2011	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE BOARD OF PATENT APPEALS
AND INTERFERENCES

Ex parte JOHN G. BELTRAN, PHILLIP D. BEYMER, and DAVID STROUD

Appeal 2009-009008
Application 10/781,307
Technology Center 2100

Before LANCE LEONARD BARRY, HOWARD B. BLANKENSHIP, and THU A. DANG, *Administrative Patent Judges*.

DANG, *Administrative Patent Judge*.

DECISION ON APPEAL

I. STATEMENT OF THE CASE

Appellants appeal under 35 U.S.C. § 134(a) from a Final Rejection of claims 1-18. We have jurisdiction under 35 U.S.C. § 6(b).

We affirm-in-part.

A. INVENTION

Appellants' invention relates to a system and method for displaying static and dynamic properties of an object, wherein a property source instance creates the dynamic properties and a property inspector generates a display window using ActiveX controls to display the properties (Abstract; ¶¶ [0036], [0043]-[0045], and [0048]). In the alternative, the objects may provide their own user interface controls in ActiveX that customize the graphical user interface of the display window on a per-property basis, wherein the user can make changes to properties that are immediately reflected in the display window (¶¶ [0055]-[0057]).

B. ILLUSTRATIVE CLAIM

Claim 1 is exemplary:

1. A computer-implemented method for displaying per-instance dynamic properties of an object comprising:

    (a) receiving a reference to an object instance having a dynamic property that is created at runtime for the object instance on a per-instance basis and is not stored with the object;

    (b) retrieving a reference to a property source instance from an association between the object and the property source instance, wherein the property source instance creates and

supplies the dynamic property and an initial value for the dynamic property for/to the object instance; and

(c) providing the reference to the object instance and the reference to the property source instance to a control, wherein the control is configured to:

(i) retrieve the dynamic property from the property source instance; and

(ii) display the dynamic property in a user interface.

C. REJECTION

The prior art relied upon by the Examiner in rejecting the claims on appeal is:

Hirsch	US 6,915,301 B2	Jul. 05, 2005 (filed on Aug. 25, 1998)

Claims 1-18 stand rejected under 35 U.S.C. 103(a) as being unpatentable over Hirsch.

II. ISSUES

The issues are whether the Examiner has erred in determining that Hirsch teaches or would have suggested:

1. "wherein the *property source instance creates* and supplies *the dynamic property*" and "providing the reference to the object instance and the reference to the property source instance to *a control*" (claim 1, emphasis added);

2. "wherein *the first object provides a custom ActiveX control* that defines a first user interface for displaying and editing the first property"

(claim 9, emphasis added) and "one of the property editors comprises a custom ActiveX control *specified by one of the objects*" (claim 13, emphasis added);

3. "wherein the *stock ActiveX control is not provided by one of the objects* that contains the one or more additional properties" (claim 14, emphasis added); and

4. "further comprising an application programming interface configured to provide *the ability to push the one or more object to the property inspector for display*" (claim 16, emphasis added).

III. FINDINGS OF FACT

The following Findings of Fact (FF) are shown by a preponderance of the evidence.

Hirsch

1. Hirsch is directed to business intelligence tools for building applications on a Database Management System (DBMS) having object oriented programming that supports objects with dynamic properties which may be edited using a property entry sheet (col. 1, ll. 5-7; col.1, ll. 45-55), wherein objects are bound to data sources (col. 11, ll. 37-43).

2. A developer may interactively build a virtual world, whose building blocks include scenes, data sources, global parameters, and resources; wherein, a scene is a visual display of information similar to a presentation slide linked to data stored in a database (col. 4, ll. 11-17).

3. Within a scene, the system displays values of a data source graphically as user-defined data elements; wherein the data sources are built

with a block diagramming tool which generates one or more database queries (col. 4, ll. 17-22).

4. A data element is the graphical representation of a row resulting from a query always associated with a data source and may contain any combination of objects (col. 4, ll. 44-51).

5. Objects within a scene are described by a set of static and dynamic properties, wherein the dynamic properties are designed as functional expressions that are entered by the user and evaluated at run time (col. 4, ll. 30-32; col. 11, ll. 37-57; and col. 12, ll. 15-17). An object inspector generates an object inspector window that displays a property entry sheet which enables the user to enter the property name and functional expression which may be referenced at run-time as a data value (Fig. 2; col. 1, ll. 45-55; col. 11, ll. 37-61). The functional expression may include the names of one or more data source columns which are automatically bound to a result row at runtime (col. 11, ll. 40-44).

6. When a property value is of an enumerated type, where the value must represent a selection from a finite number of values, a drop-down combobox may be displayed listing the legal values for the property (col. 12, ll. 17-28).

7. When the property value is a date or time, the object inspector window may use a calendar control to display a calendar when the field is active (col. 1, ll. 11-14 and col. 12, ll. 29-31).

8 . The byte code representing the virtual world may be executed by a runtime control such as an ActiveX control (col. 4, ll. 26-28).

IV. ANALYSIS

Claims 1-8, and 17

Appellants do not provide separate arguments with respect to independent claims 1, 5, and 17 (App. Br. 13-19).

In the Appeal Brief, as to claim 1, Appellants contend that "Hirsch completely and entirely fails to teach, describe, or suggest, explicitly or implicitly the creation of a property source instance for a property of a separate object instance" (App. Br. 14). Appellants assert that "the claims explicitly provide for a property source instance creating and supplying both a dynamic property and an initial value for the dynamic property to a separate object instance" (App. Br. 15). Appellants further argue that "Hirsch's dynamic properties already exist for an object [and] are merely evaluated at run time" (App. Br. 14). Appellants finally contend that "[n]owhere in Hirsch is there even a remote reference to a control receiving both a reference to an object instance and a property source instance, and then retrieving the dynamic property from the property source instance and displaying such a property" (App. Br. 15).

The Examiner, however, finds that "the claim language of Claim 1 does not reflect the creation of a property source instance" (Ans. 14). The Examiner finds further that "[d]ata elements consist of objects … , whose properties contain values that are set at runtime based on calculated expressions, making them dynamic properties" (Ans. 16). The Examiner notes that "Hirsch discloses an object inspector window [that] displays the object, the objects properties, and receives the object properties from the data source" (Ans. 16).

The Examiner finds further that "[t]he execution of a data source results in the creation of a data element, which includes objects whose properties contain values that are set at runtime based on calculated expressions" (Ans. 16). Thus, the Examiner concludes that the "retrieval of this information by the object inspector window to display the information provides the teaching of a reference between the property source (data source) and the object instances (objects)" (*id.*).

In the Reply Brief, Appellants correct the first argument in the Appeal Brief (App. Br. 14) by contending that "Hirsch completely and entirely fails to teach, describe, or suggest, explicitly or implicitly the creation *by* a property source instance of a dynamic property of a separate object instance" (Reply Br. 14, emphasis added).

Appellants' argument that Hirsch does not teach or suggest "creation *of* a property source instance" (App. Br. 14, emphasis added) is not commensurate in scope with the specific language of claim 1. In particular, claim 1 does not recite such "creation of a property source instance" as Appellants argue. Thus, we determine on this Appeal whether Hirsch discloses "wherein the property source instance creates and supplies the dynamic property" as specifically required by claim 1.

Hirsch is directed to a business intelligence tool for building applications on a Database Management System (DBMS) having object oriented programming that supports objects with dynamic properties (FF 1). These objects may be edited using a property entry sheet and are bound to data sources (FF 1). A developer may interactively build a virtual world, whose building blocks include scenes, data sources, global parameters, and resources; wherein, a scene is a visual display of information likened to a

presentation slide where the information is linked to data stored in a database (FF 2). Within a scene, the system displays values resulting from a data source graphically as user-defined data elements; wherein, *data sources comprise one or more database queries* (FF 3). A *data element is the graphical representation of a row resulting from one of these data source queries* and may contain any combination of objects (FF 4).

An object inspector window displays a property entry sheet which *enables the user to enter the* property name and *functional expression* which may be referenced at run-time as a data value (FF 5). An expression may include the names of one or more data source columns which are automatically bound to a result row at runtime (FF 5)

Although data elements are always associated with data sources and data elements, which may include objects, we find that Hirsch does not teach or would not have suggested that the data source or data element creates the dynamic properties of the object; rather, it is *the user that inputs the dynamic functional expressions for the dynamic properties of an object.*

Accordingly, we find that Appellants have shown that the Examiner erred in rejecting claims 1, 5, and 17 and claims 2-4 and 6-8 depending respectively from claims 1 and 5 under 35 U.S.C. § 103(a) as being unpatentable over Hirsch.

Claims 9-12 and 18

As for independent claim 9, Appellants contend that "nowhere in Hirsch is there any description that the object itself provides the custom control to display the properties of the object. . . . [rather] Hirsch explicitly teaches a defined set of controls that are used [by the object inspector] independent of the object." (App. Br. 22).

The Examiner, however, finds that "Hirsch discloses an object inspector window that displays object properties" wherein "the user may enter property values that may be constants or calculated values containing functions of parameters of column names from a data source," "a drop down combo-box," or "a calendar control (custom control)" (Ans. 19-20). The Examiner concludes that "[b]ased on the broadest, reasonable interpretation, the objects provide these custom controls" since "[a]n object that does not include a date or time property would not provide the calendar control when inspected by the object inspector window, giving the objects the ability to 'provide' custom controls based on which object is being inspected." (Ans. 20).

As noted *supra*, Hirsch is directed to a business intelligence tool for building applications on a Database Management System (DBMS) having object oriented programming that supports objects with dynamic properties which may be edited using a property entry sheet (FF 1). An object inspector window displays the property entry sheet which *enables the user to enter the* property name and *functional expression* which may be referenced at run-time as a data value (FF 5). The object inspector may generate a drop-down combobox when a property value must be selected from a finite number of values (FF 6) or use a calendar control when the property value is a date or time (FF 7).

We find that it is the *object inspector that provides the controls* necessary for generating the object inspector window which may include a dropdown combobox or a calendar. We, however, do not find that Hirsch discloses that the *objects* provide their own user interface controls in

ActiveX that customize the graphical user interface of the object inspector window on a per-property basis.

Accordingly, we find that Appellants have shown that the Examiner erred in rejecting independent claim 9 and claims 10-12 and 18 depending from claim 9 under 35 U.S.C. § 103(a) over Hirsch.

Claim 13

Appellants do not provide a separate argument for claim 13 since the argument for claims 9 and 13 only recite the claim language of claim 9 (App. Br. 21-24; Reply Br. 24-27).

The Examiner, however, finds that "Hirsch discloses one or more property editors wherein: (i) one of the property editors comprises a custom control specified by one of the objects, but does not explicitly disclose the control is an ActiveX control" (Ans. 9). The Examiner finds that "Hirsch does disclose supporting ActiveX controls" (*id.*).

To determine whether Hirsch would have suggested providing a "one of the property editors comprises a custom ActiveX control *specified by* one of the objects" as required by claim 13, we give the claim its broadest reasonable interpretation consistent with the Specification. *See In re Morris*, 127 F.3d 1048, 1054 (Fed. Cir. 1997). However, we will not read limitations from the Specification into the claims. *In re Van Geuns*, 988 F.2d 1181, 1184 (Fed. Cir. 1993).

Claim 13 does not place any limitation on what "specified" means, includes, or presents. Thus, we give the claim its broadest reasonable interpretation as the property editor having a *custom ActiveX control associated with an object*, as consistent with the Specification and as specifically defined in claim 13.

As noted *supra*, the object inspector window displays the property entry sheet which enables the user to enter the property name and functional expression including a calendar control that displays a calendar when the field is active (FF 5). The byte code representing the virtual world may be executed by a runtime control such as *an ActiveX control* (FF 8). We find the ActiveX control to be the code that the object inspector implements to display the object inspector window, wherein the ActiveX control is *associated* with the object since it displays the properties of the object.

Accordingly, we find that Appellants have not shown that the Examiner erred in rejecting claim 13 under 35 U.S.C. § 103(a) over Hirsch.

Claims 14 and 15

Appellants provide arguments for claim 14 and do not provide a separate argument for claim 15 from which it depends (App. Br. 24-25).

Appellants contend that "it is impossible for the object inspector to teach both the stock ActiveX controls and the custom ActiveX controls since this claim provides that the stock ActiveX controls are not provided by the object containing the remaining object properties" (App. Br. 25).

The Examiner, however, finds that "[t]he object inspector provides a 2 column entry form, where a user may use a user may use a drop-down combo box (stock control) or a calendar control (custom control) to edit the value of an object" (Ans. 20-21).

Claim 14 does not place any limitation on what "wherein the stock Active X control is not provided by one of the objects that contains the one or more additional properties" means, includes, or presents. Thus, we give the claim its broadest reasonable interpretation as the objects do not provide

stock ActiveX control, as consistent with the Specification and as specifically defined in claim 14.

As noted *supra*, we find that Hirsch does not disclose that the objects provide their own user interface controls in ActiveX on a per-property basis *whether stock or custom*. Therefore, we find that the stock ActiveX controls are not provided by the object as claimed.

Accordingly, we find that Appellants have not shown that the Examiner erred in rejecting claim 14 and claim 15 depending from claim 14 under 35 U.S.C. § 103(a) over Hirsch.

Claim 16

Appellants contend that "the use of a push based model or the ability to push a first object to a second object for display is not even remotely hinted at anywhere in the cited text or remainder of Hirsch" (App. Br. 25).

The Examiner, however, finds that "Hirsch discloses the ability to push the first object to a second object for display"; wherein,"[o]bjects (first object) and their properties are displayed in a property inspector window (second object)" (Ans. 21).

Claim 16 does not place any limitation on what "push" means, includes, or presents. Thus, we give the claim its broadest reasonable interpretation as an application programming interface (API) configured to provide *the ability to transmit objects* to the property inspector for display, as consistent with the Specification and as specifically defined in claim 16.

As noted *supra*, the object inspector generates the object inspector window that displays the property entry sheet corresponding to an object (FF 5). Accordingly, we find that the API disclosed in Hirsch is configured to

transmit objects to the property inspector for display in the object inspector window.

Accordingly, we find that Appellants have not shown that the Examiner erred in rejecting claim 16 under 35 U.S.C. § 103(a) over Hirsch.

V. CONCLUSION AND DECISION

The Examiner's rejection of claims 13-16 under 35 U.S.C. § 103(a) is affirmed. The Examiner's rejection of claims 1-12, 17, and 18 under 35 U.S.C. § 103(a) is reversed.

No time period for taking any subsequent action in connection with this appeal may be extended under 37 C.F.R. § 1.136(a)(1)(iv).

AFFIRMED-IN-PART

peb